VB.NET - DIRECTIVES

The VB.Net compiler directives give instruction to the compiler to preprocess the information before actual compilation starts.

All these directives begin with #, and only white-space characters may appear before a directive on a line. These directives are not statements.

VB.Net compiler does not have a separate preprocessor; however, the directives are processed as if there was one. In VB.Net the compiler directives are used to help in conditional compilation. Unlike C and C++ directives, they are not used to create macros.

Compiler Directives in VB.Net

VB.Net provides the following set of compiler directives:

- The #Const Directive
- The #ExternalSource Directive
- The #If...Then...#Else Directives
- The #Region Directive

The #Const Directive

This directive defines conditional compiler constants. Syntax for this directive is:

```
#Const constname = expression
```

Where,

- ***constname***: specifies the name of the constant. Required.
- ***expression***: it is either a literal, or other conditional compiler constant, or a combination including any or all arithmetic or logical operators except **Is**

For example,

```
#Const state = "WEST BENGAL"
```

Example

The following code demonstrates a hypothetical use of the directive:

```
Module mydirectives
#Const age = True
Sub Main()
   #If age Then
      Console.WriteLine("You are welcome to the Robotics Club")
   #End If
   Console.ReadKey()
End Sub
End Module
```

When the above code is compiled and executed, it produces following result:

```
You are welcome to the Robotics Club
```

The #ExternalSource Directive

This directive is used for indicating a mapping between specific lines of source code and text external to the source. It is used only by the compiler and the debugger, has no effect on code compilation.

This directive allows including external code from an external code file into a source code file.

Syntax for this directive is:

```
#ExternalSource( StringLiteral , IntLiteral )
    [ LogicalLine ]
#End ExternalSource
```

The parameters of #ExternalSource directive are the path of external file, line number of the first line, and the line where the error occurred.

Example

The following code demonstrates a hypothetical use of the directive:

```
Module mydirectives
    Public Class ExternalSourceTester

        Sub TestExternalSource()

        #ExternalSource("c:\vbprogs\directives.vb", 5)
            Console.WriteLine("This is External Code. ")
        #End ExternalSource

        End Sub
    End Class

    Sub Main()
        Dim t As New ExternalSourceTester()
        t.TestExternalSource()
        Console.WriteLine("In Main.")
        Console.ReadKey()

    End Sub
```

When the above code is compiled and executed, it produces following result:

```
This is External Code.
In Main.
```

The #If...Then...#Else Directives

This directive conditionally compiles selected blocks of Visual Basic code.

Syntax for this directive is:

```
#If expression Then
   statements
[ #ElseIf expression Then
   [ statements ]
...
#ElseIf expression Then
   [ statements ] ]
[ #Else
```

```
   [ statements ] ]
#End If
```

For example,

```
#Const TargetOS = "Linux"
#If TargetOS = "Windows 7" Then
   ' Windows 7 specific code
#ElseIf TargetOS = "WinXP" Then
   ' Windows XP specific code
#Else
   ' Code for other OS
#End if
```

Example

The following code demonstrates a hypothetical use of the directive:

```
Module mydirectives
#Const classCode = 8

   Sub Main()
   #If classCode = 7 Then
        Console.WriteLine("Exam Questions for Class VII")
   #ElseIf classCode = 8 Then
        Console.WriteLine("Exam Questions for Class VIII")
   #Else
        Console.WriteLine("Exam Questions for Higher Classes")
   #End If
        Console.ReadKey()

    End Sub
End Module
```

When the above code is compiled and executed, it produces following result:

```
Exam Questions for Class VIII
```

The #Region Directive

This directive helps in collapsing and hiding sections of code in Visual Basic files.

Syntax for this directive is:

```
#Region "identifier_string"
#End Region
```

For example,

```
#Region "StatsFunctions"
    ' Insert code for the Statistical functions here.
#End Region
```